Computability 7 (2018) 259–271
DOI 10.3233/COM-170073
IOS Press

A probabilistic anytime algorithm for the halting problem

Cristian S. Calude
Department of Computer Science, University of Auckland, Auckland, New Zealand
cristian@cs.auckland.ac.nz
www.cs.auckland.ac.nz/~cristian

Monica Dumitrescu
Faculty of Mathematics and Computer Science, Bucharest University, Romania
mdumi@fmi.unibuc.ro
http://goo.gl/txsqpU

The first author dedicates his contribution to this paper to the memory of his collaborator and friend S. Barry Cooper (1943–2015).

Abstract. The Halting Problem, the most (in)famous undecidable problem, has important applications in theoretical and applied computer science and beyond, hence the interest in its approximate solutions.

Experimental results reported on various models of computation suggest that halting programs are not uniformly distributed – running times play an important role. A reason is that a program which eventually stops but does not halt "quickly", stops at a time which is algorithmically compressible.

In this paper we work with running times to define a class of computable probability distributions on the set of halting programs in order to construct an anytime algorithm for the Halting problem with a probabilistic evaluation of the error of the decision.

Keywords: Halting Problem, anytime algorithm, running time distribution

1. Introduction

The Halting Problem asks to decide, from a description of an arbitrary program and an input, whether the computation of the program on that input will eventually stop or continue forever. In 1936 Alonzo Church, and independently Alan Turing, proved that (in Turing's formulation) an *algorithm to solve the Halting Problem for all possible program-input pairs does not exist*; two equivalent models have been used to describe computation by algorithms (an informal notion), the lambda calculus by Church and Turing machines by Turing. The Halting Problem is historically the first proved undecidable problem; it has many applications in mathematics, logic and theoretical as well as applied computer science, mathematics, physics, biology, etc. Due to its practical importance approximate solutions for this problem have been proposed for quite a long time, see [2,5–7,9,14,16,19,21,26].

Anytime algorithms trade execution time for quality of results [13]. These algorithms can be executed in two modes: either by a given contract time to execute or an interruptible method. Instead of correctness, an anytime algorithm returns a result together with a "quality measure" which evaluates how close the obtained result is to the result that would be returned if the algorithm ran until completion (which may be prohibitively long). To improve the solution, anytime algorithms can be continued after they have halted. That is similar to the use of iterative processes in numerical computing in which, after the process has been halted, if the output is not considered to be acceptable, then it can be refined by resuming the iterative process.

Following Manin [21] we use a more general form of anytime algorithm as an approximation for a computation which may never stop. An anytime algorithm for the Halting Problem works in the following way: to test whether a program eventually stops on a given input we first *effectively compute a threshold time* – the interruptible (stopping) condition – and then run the program for that specific time. If the computation stops, then the program was proved to halt; if the computation does not stop, then we *declare* that: (a) the program will never stop and (b) evaluate the

probability of error, i.e. the probability that the program may eventually stop. The goal is to prove that the *probability of error* can be made as small as we wish. By running the program a longer time we can improve its performance either by getting to the halting time or by decreasing the probability of error. Another important goal is to develop feasible anytime algorithms for the Halting Problem.

In [6,7] anytime algorithms for the Halting Problem have been developed using the fact – proved in [7] – that programs which take a long time to halt stop at "algorithmically compressible times", i.e. times which a computer can generate from "smaller" inputs. Although the set of algorithmically compressible times has constructive density zero, the stopping times obtained using this method are very large and the theoretical bounds cannot be improved in general [6]. However, experimental results in [30] for small Turing machines indicate much smaller stopping times. Furthermore, the experimental results in [27,30,31] and theoretical results in [14,15] show that halting programs *are not uniformly distributed* and their distributions depend on the running times of the specific model of computation.

In this paper we construct a class of computable probability distributions on the set of halting programs based on their running times. Each computable probability distribution induces a probability space on the set of halting programs. Using this probabilistic framework we construct an anytime algorithm for the Halting problem with a probabilistic evaluation of the error of the decision.

The paper is organised as follows. We start with a section on basic notation. Section 3 presents the computability and complexity part while Section 4 is dedicated to probability and statistics. Section 5 is dedicated to the presentation of the probabilistic framework for the anytime algorithms. We start by examining two plausible *a priori* halting probabilities. The analysis of their inadequacy leads to the introduction of running time probability spaces on the set of halting programs which, as their names indicate, are constructed using computable probability distributions on the set of running times. In Section 6 describe the anytime algorithm and prove its correctness. In Section 8 we propose methods to improve the performance of the algorithm and evaluate it power and limits. Finally, we discuss some open problems.

2. Notation

In the following we will denote by $\mathbb{Z}^+$ the set of positive integers $\{1, 2, \dots\}$ and let $\overline{\mathbb{Z}^+} = \mathbb{Z}^+ \cup \{\infty\}$; $\mathbb{R}$ is the set of reals. The domain of a partial function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is denoted by $\mathrm{dom}(F)$: $\mathrm{dom}(F) = \{x \in \mathbb{Z}^+ \mid F(x) < \infty\}$. We denote by $\#S$ the cardinality of the set S and by $\mathcal{P}(X)$ the power set of X.

We assume familiarity with elementary computability theory and algorithmic information theory [4,11,20].

For a partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ we denote by $F(x)[t] < \infty$ the statement "the algorithm computing F has stopped *exactly* in time t". For $t \in \mathbb{Z}^+$ we consider the computable set $\mathrm{Stop}(F, t) = \{x \in \mathbb{Z}^+ \mid F(x)[t] < \infty\}$, and note that

$$\mathrm{dom}(F) = \bigcup_{t \in \mathbb{Z}^+} \mathrm{Stop}(F, t). \tag{1}$$

3. Complexity and universality

The *algorithmic complexity* relative to a partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ is the partial function $\nabla_F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ defined by $\nabla_F(x) = \inf\{y \in \mathbb{Z}^+ \mid F(y) = x\}$. If $F(y) \neq x$ for every $y \geqslant 1$, then $\nabla_F(x) = \infty$. That is, the algorithmic complexity of x is the smallest description/encoding of x with respect to the interpreter/decoder F, or infinity if F cannot produce x.

A partially computable function $\mathbf{U}$ is called *universal* if for every partially computable function $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ there exists a constant $k_{\mathbf{U},F}$ such that for every $x \in \mathrm{dom}(\nabla_F)$ we have

$$\nabla_{\mathbf{U}}(x) \leqslant k_{\mathbf{U},F} \cdot \nabla_F(x). \tag{2}$$

Theorem 3.1 ([6]). *A partially computable function* $\mathbf{U}$ *is universal iff for every partially computable function* $F\colon \mathbb{Z}^+ \longrightarrow \overline{\mathbb{Z}^+}$ *there exists a constant* $c_{\mathbf{U},F}$ *such that for every* $x \in \mathrm{dom}(F)$ *we have*

$$\nabla_{\mathbf{U}}\big(F(x)\big) \leqslant c_{\mathbf{U},F} \cdot x. \tag{3}$$

The difference between (2) and (3) is in the role played by F: in the traditional condition (2), F appears through ∇_F (which for some F can be incomputable), while in (3) F appears as argument of $\nabla_{\mathbf{U}}$, making the second member of the inequality always computable.

A universal partially computable function $\mathbf{U}$ "simulates" any other partially computable function F in the following sense: if $x \in \mathrm{dom}(F)$, then from (3) we deduce that $\nabla_{\mathbf{U}}(F(x)) \leqslant c_{\mathbf{U},F} \cdot x$, hence there exists an $y \leqslant c_{\mathbf{U},F} \cdot x$ in $\mathrm{dom}(\mathbf{U})$ such that $\mathbf{U}(y) = F(x)$. In particular, $\nabla_{\mathbf{U}}(x) < \infty$, for all $x \in \mathbb{Z}^+$.

The set $\mathrm{dom}(\mathbf{U})$ (see (1) for $\mathbf{U} = F$) is computably enumerable, but not computable (the *undecidability of the Halting Problem*), its complement $\overline{\mathrm{dom}(\mathbf{U})}$ is not computably enumerable, but the sets $(\mathrm{Stop}(\mathbf{U}, t))_{t\geqslant 1}$ are all computable.

To solve the Halting Problem means to determine for an arbitrarily pair (F, x), where F is a partially computable function and $x \in \mathbb{Z}^+$, whether $F(x)$ stops or not, or equivalently, whether $x \in \mathrm{dom}(F)$, that is, $x \in \mathrm{Stop}(F, t)$, for some $t \in \mathbb{Z}^+$. In view of (2) or (3) solving the Halting Problem for a fixed universal $\mathbf{U}$ is enough to solve the Halting Problem. From now on we fix a universal $\mathbf{U}$ and study the Halting Problem $\mathbf{U}(x) < \infty$, for $x \in \mathbb{Z}^+$.

4. A glimpse of probability theory

In this section we define the main notions from probability theory used in this paper. For more details see [10,25].

A *measurable space* $(\Omega, \mathcal{B}(\Omega))$ consists of a non-empty set Ω and a Borel field of subsets of Ω, $\mathcal{B}(\Omega) \subseteq \mathcal{P}(\Omega)$. A *probability space* is a triple $(\Omega, \mathcal{B}(\Omega), \mathrm{Pr})$, where $(\Omega, \mathcal{B}(\Omega))$ is a measurable space and $\mathrm{Pr}\colon \mathcal{B}(\Omega) \longrightarrow [0, 1]$ is a probability measure, that is, Pr satisfies the following two conditions: (a) the probability of a countable union of mutually-exclusive sets in $\mathcal{B}(\Omega)$ is equal to the countable sum of the probabilities of each of these sets, and (b) $\mathrm{Pr}(\Omega) = 1$. We interpret $\mathcal{B}(\Omega)$ as "the family of events" and Ω as "the certain event".

Consider a probability space $(\Omega, \mathcal{B}(\Omega), \mathrm{Pr})$ and a measurable space $(A, \mathcal{B}(A))$. A *random variable* is a measurable function $X\colon \Omega \longrightarrow A$, that is, for every $B \in \mathcal{B}(A)$ we have $X^{-1}(B) \in \mathcal{B}(\Omega)$. In this case X induces a probability (called *probability distribution of* X) $P_X : \mathcal{B}(A) \longrightarrow [0, 1]$ defined by

$$P_X(B) = \mathrm{Pr}\big(X^{-1}(B)\big) = \mathrm{Pr}\big(\{\omega \mid X(\omega) \in B\}\big), \quad B \in \mathcal{B}(A),$$

which identifies the probability space $(A, \mathcal{B}(A), P_X)$.

The random variable X has a *discrete probability distribution* if A is at most countable. If we denote by $P_X(\{x\})$ the probability of the event $\{X = x\} = \{\omega \in \Omega \mid X(\omega) = x\}$, then the discrete probability distribution of X is completely defined by the numbers $P_X(\{x\}) \in [0, 1]$, $x \in A$, with $\sum_{x\in A} P_X(\{x\}) = 1$. A computable probability distribution P_X is a discrete probability distribution such that the function $x \in A \hookrightarrow P_X(\{x\})$ is computable (in particular, $P_X(\{x\})$ is a computable real for each $x \in A$ [23, p. 159]; see also [24,29]).

In what follows we assume that $A \subseteq \mathbb{R}$.

The *Cumulative Distribution Function* of a random variable X is the function $\mathrm{CDF}_X : \mathbb{R} \longrightarrow [0, 1]$ defined by $\mathrm{CDF}_X(y) = \mathrm{Pr}(X \leqslant y)$, $y \in \mathbb{R}$. In case X is a random variable with a discrete distribution, CDF_X is the stair-function (with piecewise-constant sections) given by

$$\mathrm{CDF}_X(y) = \sum_{\{x\in A \mid x\leqslant y\}} P_X(x), \quad y \in \mathbb{R}.$$

For example, the generally accepted model for "the time-to-the-first-success" is the *geometric distribution* – the discrete probability distribution that expresses the probability that the first occurrence of an event ("success") requires k independent trials, each with the same success probability θ. More precisely, the random variable X that

takes values in $A = \mathbb{Z}^+$ has a geometric distribution with the rate of success $\theta \in (0, 1)$ if $P_X(k) = (1-\theta)^{k-1} \cdot \theta$, $k \geqslant 1$. In this case

$$\mathrm{CDF}_X(y) = \sum_{k=1, k \leqslant y}^{\infty} (1-\theta)^{k-1} \cdot \theta, \quad y \in \mathbb{R},$$

and $\lim_{y \to \infty} \mathrm{CDF}_X(y) = 1$.

The *Quantile Function* of the random variable X with a discrete distribution is the function $\mathbf{q}_X : [0, 1] \longrightarrow A$ defined by $\mathbf{q}_X(p) = \inf\{y \in A \mid p \leqslant \mathrm{CDF}_X(y)\}$. By definition, $\mathrm{CDF}_X(\mathbf{q}_X(p)) \geqslant p$, for all $p \in [0, 1]$.

For fixed $r \in [0, 1]$, the value (number) $\mathbf{q}_X(r)$ is called *the* rth *quantile* of the random variable X. Quantiles are important indicators that give information about the location and clustering of the probability values $\{P_X(x), x \in A\}$. For example, if the data being studied are not actually distributed according to an assumed underlying probability distribution or if there are outliers far removed from the mean, then quantiles may provide useful information. Beside the classical quartiles – first, second (median), third – the lower and upper εth quantiles, $\mathbf{q}_X(\varepsilon)$ and $\mathbf{q}_X(1-\varepsilon)$, give important informations about the "tails" of the probability distribution (for small $\varepsilon > 0$). For more details see [1].

Proposition 4.1. *For every* $\varepsilon \in (0, 1)$ *we have* $P_X(\{x \in A \mid x > \mathbf{q}_X(1-\varepsilon)\}) \leqslant \varepsilon$, *hence* $\lim_{\varepsilon \to 0} P_X(\{x \in A \mid x > \mathbf{q}_X(1-\varepsilon)\}) = 0$.

Proof. Indeed, we have:

$$\begin{aligned} P_X\big(\{x \in A \mid x > \mathbf{q}_X(1-\varepsilon)\}\big) &= 1 - P_X\big(\{x \in A \mid x \leqslant \mathbf{q}_X(1-\varepsilon)\}\big) \\ &= 1 - \mathrm{CDF}_X\big(\mathbf{q}_X(1-\varepsilon)\big) \leqslant \varepsilon. \end{aligned}$$

□

5. A probabilistic framework

In this section we describe a probabilistic framework for developing two anytime algorithms based on an analysis of the finite running times of all halting computations $\mathbf{U}(x)$, $x \in \mathrm{dom}(\mathbf{U})$.

5.1. Halting probability

Both approaches discussed in this paper depend – directly or indirectly – on a computable probability which has to model the informal notion of "halting probability" for programs for a universal $\mathbf{U}$, that is, the probability that $\mathbf{U}$ stops on input x. *Which probability measure should we choose?*

First, various studies of concrete probability distributions for halting programs in different models of computations (see [15,27,30,31]) show that halting programs *are not uniformly distributed* and their probability distributions *depend on the running times* on the specific model of computation. This experimental evidence is reflected also in the theoretical result stating that programs which take a long time to halt stop at "algorithmically compressible times", a set of constructive density zero [7].

Second, if we interpret the event "$\mathbf{U}$ stops on input x" as a "success" in a sequence of trials, then the running time of the computation $\mathbf{U}(x)$ becomes "the-time-to-the-first-success". In this setting, the event "$\mathbf{U}(x)$ stops exactly in time t", that is "$x \in \mathrm{Stop}(\mathbf{U}, t)$", is interpreted as "$t$ is the achieved time-to-the-first success". Then, the truncated geometric distribution is a candidate for the halting probability. Is the geometric distribution probability a "natural" model for the halting probability? Can it be used for developing anytime algorithms for the Halting Problem? The answer to the first question is *negative*. Indeed, the phenomenon modelled by the geometric distribution requires a sequence of independent trials in which each trial has the same probability of success – a strong condition hardly satisfied by any $\mathbf{U}$. For the second question we note that the computability of the probability – an essential property for evaluating the stopping condition of the anytime algorithm – depends on the computability of θ and $\sum_{t \in \mathbf{T}_\mathbf{U}} (1-\theta)^{t-1}$, which may be problematic even if $\mathbf{T}_\mathbf{U}$ is computable, see [28].

This brief analysis suggests that instead of "assuming" an *a priori* halting probability we should instead construct a model for the halting probability based on the running times $\mathbf{T}_\mathbf{U}$.

5.2. A running time probability space

Recall that the *finite running times*[1] of the computations $\mathbf{U}(x)$ are the set of exact stopping times for the halting programs of $\mathbf{U}$:

$$\begin{aligned}\mathbf{T}_{\mathbf{U}} &= \{t \in \mathbb{Z}^+ \mid \text{there exists } x \in \mathbb{Z}^+ \text{ such that } x \in \text{Stop}(\mathbf{U}, t)\} \\ &= \{t \in \mathbb{Z}^+ \mid \text{there exists } x \in \mathbb{Z}^+ \text{ such that } \mathbf{U}(x)[t] < \infty\}.\end{aligned}$$

Lemma 5.1. *The set* $\mathbf{T}_{\mathbf{U}}$ *is infinite.*

Proof. The statement in the lemma is true because for every $M \in \mathbb{Z}^+$ there is a program $x \in \text{dom}(\mathbf{U})$ which stops in time larger than M: indeed, otherwise all programs would stop in time at most M, hence $\text{dom}(\mathbf{U})$ would be decidable, a contradiction. □

The undecidability of the Halting Problem casts a "computational uncertainty" on the membership problem of the set $\text{dom}(\mathbf{U})$. In what follows we model this phenomenon by introducing a probabilistic structure on $\text{dom}(\mathbf{U})$.

From (1) we have

$$\text{dom}(\mathbf{U}) = \bigcup_{t \in \mathbb{Z}^+} \text{Stop}(\mathbf{U}, t) = \bigcup_{t \in \mathbf{T}_{\mathbf{U}}} \text{Stop}(\mathbf{U}, t). \tag{4}$$

Next we note that

$$\text{Stop}(\mathbf{U}, t) \cap \text{Stop}\big(\mathbf{U}, t'\big) = \begin{cases} \emptyset, & \text{if } t \neq t', \\ \text{Stop}(\mathbf{U}, t), & \text{otherwise.} \end{cases}$$

Let us consider the family of (finite and countable) unions of sets $\text{Stop}(\mathbf{U}, t)$, $t \in \mathbb{Z}^+$. This family includes $\text{dom}(\mathbf{U})$ and is closed under complement

$$\overline{\text{Stop}(\mathbf{U}, t)} = \bigcup_{t' \in \mathbf{T}_{\mathbf{U}} \setminus \{t\}} \text{Stop}\big(\mathbf{U}, t'\big),$$

and countable unions; accordingly, it is a Borel field of subsets of $\text{dom}(\mathbf{U})$, which we denote by $\mathcal{B}(\text{dom}(\mathbf{U}))$. To define the discrete probability measure on the measurable set $(\text{dom}(\mathbf{U}), \mathcal{B}(\text{dom}(\mathbf{U})))$ we fix a computable probability distribution ρ on $\mathbf{T}_{\mathbf{U}}$ (see Section 5.3). Then, we put

$$\Pr = \Pr_{\rho} : \mathcal{B}\big(\text{dom}(\mathbf{U})\big) \longrightarrow [0, 1], \qquad \Pr\big(\text{Stop}(\mathbf{U}, t)\big) = \rho(t), \quad t \in \mathbf{T}_{\mathbf{U}}. \tag{5}$$

Now we can introduce a probability structure on the set $\mathbf{T}_{\mathbf{U}}$ via a random variable. Let $\mathcal{B}(\mathbf{T}_{\mathbf{U}})$ be the family of all subsets of $\mathbf{T}_{\mathbf{U}}$. The function

$$RT = RT_{\mathbf{U}} : \text{dom}(\mathbf{U}) \longrightarrow \mathbf{T}_{\mathbf{U}}, \qquad RT(x) = \min\{t > 0 \mid x \in \text{Stop}(\mathbf{U}, t)\} \tag{6}$$

has the property that for every $t \in \mathbf{T}_{\mathbf{U}}$, $RT^{-1}(\{t\}) = \text{Stop}(\mathbf{U}, t) \in \mathcal{B}(\text{dom}(\mathbf{U}))$. Consequently, RT is a *random variable* – in fact a stopping time – which will be called the *running time associated with* $\mathbf{U}$. As described in Section 4, the random variable RT induces the probability space $(\mathbf{T}_{\mathbf{U}}, \mathcal{B}(\mathbf{T}_{\mathbf{U}}), P_{\text{RT}})$ on $\mathbf{T}_{\mathbf{U}}$ in which the probability is defined as follows:

$$P_{\text{RT}}\big(\{t\}\big) = \Pr\big(RT^{-1}\big(\{t\}\big)\big), \quad t \in \mathbf{T}_{\mathbf{U}}.$$

[1] See [12] for modelling running times.

It is seen that for every $t \in \mathbf{T}_\mathbf{U}$:

$$P_{\mathrm{RT}}\big(\{t\}\big) = \Pr\big(\mathrm{Stop}(\mathbf{U}, t)\big) = \rho(t).$$

In what follows a computable probability space $(\mathrm{dom}(\mathbf{U}), \mathcal{B}(\mathrm{dom}(\mathbf{U})), \Pr_{\rho_\mathbf{U}})$ of the form defined in (5) will be called a *running time probability space.*

The random variable RT is completely specified by a *computable probability distribution on the set of finite running times of programs of* $\mathbf{U}$,

$$\big\{\rho(t) \mid t \in \mathbf{T}_\mathbf{U}\big\}. \tag{7}$$

Of course, we need to prove that a computable probability distribution (7) exists: this will be done in Section 5.3.

The cumulative distribution function $\mathrm{CDF}_{\mathrm{RT}} : \mathbf{T}_\mathbf{U} \longrightarrow [0, 1]$ of the discrete random variable $RT\colon\ \mathrm{dom}(\mathbf{U}) \longrightarrow \mathbf{T}_\mathbf{U}$ is then defined by the formula:

$$\mathrm{CDF}_{\mathrm{RT}}(k) = P_{\mathrm{RT}}\big(\{t \in \mathbf{T}_\mathbf{U} \mid 1 \leqslant t \leqslant k\}\big) = \sum_{t=1, t\in\mathbf{T}_\mathbf{U}}^{k} P_{\mathrm{RT}}(t), \quad k \in \mathbf{T}_\mathbf{U},$$

and the quantile function of RT is

$$\mathbf{q}_{\mathrm{RT}}(r) = \inf\big\{k \in \mathbf{T}_\mathbf{U} \mid \mathrm{CDF}_{\mathrm{RT}}(k) \geqslant r\big\}, \quad r \in (0, 1).$$

For $\varepsilon \in (0, 1)$ we now use the $(1-\varepsilon)$-quantile $\mathbf{q}_{\mathrm{RT}}(1-\varepsilon)$ as a *probabilistic threshold* separating the "the upper ε-tail" of the distribution, i.e. those very large running times t making the event "$\mathbf{U}(x)[t] < \infty$" negligible according to P_{RT}.

5.3. Running time computable probability distributions

We now discuss examples of finite running time computable probability distributions for (7).

Every computable probability on $\mathrm{dom}(\mathbf{U})$ is defined on a computably enumerable, but not computable set and has to satisfy the equality $\rho_\mathbf{U}(t) = \Pr(\mathrm{Stop}(\mathbf{U}, t))$, for all $t \in \mathbf{T}_\mathbf{U}$, so it has to depend on $\mathbf{U}$. Furthermore, to construct $\rho_\mathbf{U}$ we need:

- a computable function $\mathrm{pr}_\mathbf{U}\colon \mathbb{Z}^+ \times \mathbb{Z}^+ \to [0, 1]$ such that $\mathrm{pr}_\mathbf{U}(x, t) > 0$ iff $x \in \mathrm{Stop}(\mathbf{U}, t)$ (the probability that $U(x)$ stops on time t),
- a computable sequence of computable reals $\upsilon_\mathbf{U}(t) = \sum_{x=1}^{\infty} \mathrm{pr}(x, t)$.

If $\sum_{t=1}^{\infty} \upsilon_\mathbf{U}(t) = 1$, then we can set $\rho_\mathbf{U}(t) = \upsilon_\mathbf{U}(t) = \Pr(\mathrm{Stop}(\mathbf{U}, t))$. If $\sum_{t=1}^{\infty} \upsilon_\mathbf{U}(t) < 1$, we need to normalise with $\Upsilon_\mathbf{U} = \sum_{t=1}^{\infty} \upsilon_\mathbf{U}(t)$, hence $\Upsilon_\mathbf{U}$ has to be computable. In this case

$$\rho_\mathbf{U}(t) = \frac{1}{\Upsilon_U} \cdot \upsilon_\mathbf{U}(t) = \Pr\big(\mathrm{Stop}(\mathbf{U}, t)\big).$$

As an example we construct the computable probability distribution $\rho_\mathbf{U}$ by using the function $\mathrm{pr} = \mathrm{pr}_\mathbf{U}\colon \mathbb{Z}^+ \times \mathbb{Z}^+ \to [0, 1]$ defined by

$$\mathrm{pr}(x, t) = \begin{cases} \frac{2^{-x}}{t}, & \text{if } x \in \mathrm{Stop}(\mathbf{U}, t), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The function pr is computable because the sets $\mathrm{Stop}(\mathbf{U}, t)$ are computable for every $t \in \mathbb{Z}^+$.

Theorem 5.2. *The real numbers*

$$\upsilon(t) = \upsilon_{\mathbf{U}}(t) = \sum_{x=1}^{\infty} \mathrm{pr}(x,t), \quad t \in \mathbb{Z}^+, \tag{9}$$

$$\Upsilon = \Upsilon_{\mathbf{U}} = \sum_{t=1}^{\infty} \upsilon(t) \tag{10}$$

are computable and $0 \leqslant \upsilon(t) < \Upsilon < 1$.

Proof. To prove that $\upsilon(t)$ is computable we use Theorem 4.2.3 in [29] which says that the limit of a computable sequence of rationals having a computable modulus of convergence is a computable real. To this aim we need to prove that the modulus of convergence of the series (9) is computable. Indeed, for every positive integers $i < j$ and n we have

$$\sum_{x=1, x\in \mathrm{Stop}(\mathbf{U},t)}^{j} \frac{2^{-x}}{t} - \sum_{x=1, x\in \mathrm{Stop}(\mathbf{U},t)}^{i} \frac{2^{-x}}{t} = \sum_{x=i+1, x\in \mathrm{Stop}(\mathbf{U},t)}^{j} \frac{2^{-x}}{t}$$
$$\leqslant \sum_{x=i+1}^{\infty} \frac{2^{-x}}{t} = \frac{2^{-i}}{t} \leqslant 2^{-n},$$

for $i \geqslant \lceil \log_2(\frac{2^{-i}}{t}) \rceil$, a computable convergence modulus.

A similar argument works for the series (10) because the set $\{(t,x) \in \mathbb{Z}^+ \times \mathbb{Z}^+ \mid x \in \mathrm{Stop}(\mathbf{U},t)\}$ is computable. □

Note that if $x \notin \mathrm{dom}(\mathbf{U})$, then for every $t \in \mathbb{Z}^+$ we have $\mathrm{pr}(x,t) = 0$, so $\upsilon(t) = 0$.

Using Theorem 5.2 we construct the following *computable probability distribution* ρ on the set of finite running times $\mathbf{T}_{\mathbf{U}}$:

$$\rho(t) = \rho_{\mathbf{U}}(t) = \frac{\upsilon(t)}{\Upsilon} = \frac{1}{\Upsilon} \sum_{x=1}^{\infty} \mathrm{pr}(x,t), \quad t \in \mathbf{T}_{\mathbf{U}}. \tag{11}$$

Indeed, from (11) and (10) we have:

$$\sum_{t\in \mathbf{T}_{\mathbf{U}}} \rho(t) = \sum_{t=1}^{\infty} \rho(t) = \sum_{t=1}^{\infty} \frac{1}{\Upsilon} \sum_{x=1}^{\infty} \mathrm{pr}(x,t) = 1.$$

As described in Section 4, using ρ we define the computable discrete probability space $(\mathbf{T}_{\mathbf{U}}, \mathcal{B}(\mathbf{T}_{\mathbf{U}}), P_\rho)$, where $\mathcal{B}(\mathbf{T}_{\mathbf{U}})$ is the set of all subsets of $\mathbf{T}_{\mathbf{U}}$ and $P_\rho(t) = \rho(t)$.

The series (9) is a semi-measure [18, Section 4] with a computable sum[2] – by Theorem 5.2, (10) – an essential property for the computability of the probability P_ρ.

The above probability space – inspired from [7] – is "natural" because the discrete probability distribution combines the uniform distribution assumed in the halting probability Ω number, see [4], with the time complexity of halting programs (normalised by the computable number Υ, see (10)). In detail, the function (8) biases the programs x – assumed to be uniformly distributed – by dividing 2^{-x} to the program's stopping time t.

[2] The sum of a computable semi-measure may be not computable as Specker theorem [28] indicates.

A program which eventually stops but does not halt "quickly" stops at an algorithmically compressible time, hence the probability of a program, that doesn't stop for a long time, to halt tends to zero, see [6,7]. More precisely, if $x \in \text{Stop}(\mathbf{U}, t)$, then the longer t is, the smaller the halting probability of x is; if the program never halts, that is $x \notin \text{Stop}(\mathbf{U}, t)$, for all t, then the halting probability of x tends to zero when $t \to \infty$. Any computable probability distribution not reflecting this phenomenon is "un-natural".

To further justify the "naturalness" of the probability P_ρ we now show that it reflects the behaviour of *both* halting and non-halting programs. To this aim we use the series (9) to define a variation of ρ, namely a semi-computable probability distribution r on the set of all running times, finite or infinite, $\mathbf{T}_\mathbf{U} \cup \{\infty\}$ as follows:

$$r(t) = \begin{cases} \upsilon(t), & \text{if } t \in \mathbf{T}_\mathbf{U}, \\ 1 - \Upsilon, & t = \infty. \end{cases}$$

As by (10) and (11)

$$\sum_{t=1}^{\infty} r(t) + r(\infty) = \sum_{t=1}^{\infty} \sum_{x=1}^{\infty} \text{pr}(x, t) + r(\infty) = 1,$$

we can define $P_r(\{t\}) = r(t)$ for $t \in \mathbf{T}_\mathbf{U} \cup \{\infty\}$ to obtain the semi-computable probability space $(\mathbf{T}_\mathbf{U} \cup \{\infty\}, \mathcal{B}(\mathbf{T}_\mathbf{U} \cup \{\infty\}), P_r)$, where $\mathcal{B}(\mathbf{T}_\mathbf{U} \cup \{\infty\})$ is the set of all subsets of $\mathbf{T}_\mathbf{U} \cup \{\infty\}$.

In contrast with P_ρ – which deals only with finite running times – P_r handles also the infinite running time, the running time of non-halting programs. The normalisation factor Υ makes P_ρ "reflect" the behaviour of non-halting programs too as the restriction of P_r to $\mathbf{T}_\mathbf{U}$ is

$$P_\rho(t) = \frac{P_r(\{t\})}{\Upsilon}, \quad t \in \mathbf{T}_\mathbf{U}.$$

In (8) t can be replaced with $\log(t + 1)$ or, more generally, with $g(t)$, where g is a non-decreasing, unbounded, computable function: in this way we obtain a class of computable probabilistic distributions on $\mathbf{T}_\mathbf{U}$.

In what follows, a computable probability distribution ρ on $\mathbf{T}_\mathbf{U}$ will be extended to $\mathbb{Z}^+$ by setting $\rho(t) = 0$ for every $t \in \mathbb{Z}^+ \setminus \mathbf{T}_\mathbf{U}$.

Finally, we note that the problem whether a concrete computable probability distribution is "natural" depends on various factors, some objective, others subjective. There are a few ways to mitigate subjectivity, in particular, to make (11) more "practical". One possibility is to use instead of t a very slow increasing computable function $g(t)$. A more substantial improvement can be obtained using Proposition 1.5.2 in [22]. Yet another way will be discussed in Section 7.

6. The probabilistic anytime algorithm

As we mentioned in Section 3, to solve the Halting Problem is enough to fix a universal $\mathbf{U}$ and to decide, for an arbitrary program x, whether $\mathbf{U}(x) < \infty$ or $\mathbf{U}(x) = \infty$.

Our aim is to construct tan anytime algorithm for testing the incomputable predicate "$\mathbf{U}(x) < \infty$". The decision to accept/reject the hypothesis "$\mathbf{U}(x) < \infty$" will be based on the running time of the computation $\mathbf{U}(x)$. A decision made by the anytime algorithm is *erroneous* when it returns the output "$\mathbf{U}(x) = \infty$", when, in fact, $\mathbf{U}(x) < \infty$ (that is, $\mathbf{U}(x)$ eventually stops after a very long time).

The Halting Problem will be re-formulated within the probabilistic framework presented in Section 5 as follows:

For arbitrary $x \in Z^+$, test the hypothesis $H_x : \{\mathbf{U}(x) < \infty\}$ against the alternative $H'_x : \{\mathbf{U}(x) = \infty\}$.

The decision of rejecting H_x will be taken on the basis of a *critical time region* B_x. In both proposed anytime algorithms, the critical regions will *not depend on* x, that is, $B = B_x$, for every $x \in \mathbb{Z}^+$.

An erroneous decision occurs when we reject H_x on the basis of B, but H_x is true. The quality of this decision is expressed by the probability of an erroneous decision, i.e. the probability that a halting program x stops in a time $t \in B$.

In what follows we will work with an a priori running time probability space $(\mathbf{T_U}, \mathcal{B}(\mathbf{T_U}), P_{\mathrm{RT}})$ defined in (7) and the running time random variable RT defined in (6). Our main example is $P_{\mathrm{RT}} = P_\rho$, where ρ comes from (8).

In what follows we fix a computable probability distribution P_{RT}.

First we apply Proposition 4.1 to the random variable RT:

Corollary 6.1. *For every $\varepsilon \in (0, 1)$ we have*

$$P_{\mathrm{RT}}\big(\{t \in \mathbf{T_U} \mid t > \mathbf{q}_{\mathrm{RT}}(1-\varepsilon)\}\big) \leqslant \varepsilon, \tag{12}$$

hence $\lim_{\varepsilon \to 0} P_{\mathrm{RT}}(\{t \in \mathbf{T_U} \mid t > \mathbf{q}_{\mathrm{RT}}(1-\varepsilon)\}) = 0$.

We now use the inequality (12) in Corrolary 6.1 to propose the following *probabilistic anytime algorithm* for the Halting Problem:[3]

Fix $\varepsilon = \frac{1}{M}$ with $M \in \mathbb{Z}^+$. Let x be an arbitrary program for $\mathbf{U}$. If the computation $\mathbf{U}(x)$ does not stop in time less than or equal to $\mathbf{q}_{\mathrm{RT}}(1-\varepsilon)$, then declare that $\mathbf{U}(x) = \infty$.

If the computation $\mathbf{U}(x)$ stops in time less than or equal to $\mathbf{q}_{\mathrm{RT}}(1-\varepsilon)$, then obviously $\mathbf{U}(x) < \infty$. Otherwise, the answer to the question whether $\mathbf{U}(x) < \infty$ is *unknown* and *algorithmically unknowable*. The above anytime algorithm gives an approximate answer. To analyse the quality of the answer produced by this anytime algorithm we choose the *computable* critical time region[4]

$$\mathbf{B}(P_{\mathrm{RT}}, \varepsilon) = \big\{t \in \mathbf{T_U} \mid t > \mathbf{q}_{\mathrm{RT}}(1-\varepsilon)\big\},$$

and the *critical program region*

$$\begin{aligned}\mathbf{C}(P_{\mathrm{RT}}, \varepsilon) &= \big\{x \in \mathbb{Z}^+ \mid \mathbf{U}(x)[t] = \infty, \text{for some } t \in \mathbf{B}(P_{\mathrm{RT}}, \varepsilon)\big\} \\ &= \big\{x \in \mathrm{dom}(\mathbf{U}) \mid RT(x) \in \mathbf{B}(P_{\mathrm{RT}}, \varepsilon)\big\}.\end{aligned}$$

Note that

$$\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U}) \subseteq \mathbf{C}(P_{\mathrm{RT}}, \varepsilon) \subset \mathbb{Z}^+.$$

The first inclusion above is not necessarily an equality as there may exist $t_1 > t_0 > \mathbf{q}_{\mathrm{RT}}(1-\varepsilon)$ such that $\mathbf{U}(x)[t_0] = \infty$ and $\mathbf{U}(x)[t_1] < \infty$.

The anytime algorithm may output the answer "$\mathbf{U}(x) = \infty$" when in fact $\mathbf{U}(x) < \infty$. To evaluate the quality of the anytime algorithm we need to "compare" the set $\mathbf{C}(P_{\mathrm{RT}}, \varepsilon)$ – which gives the "anytime" answers "$\mathbf{U}(x) = \infty$" – with the exact set $\mathbb{Z}^+ \setminus \mathrm{dom}(\mathbf{U})$ – giving the correct answers "$\mathbf{U}(x) = \infty$". To this aim we evaluate the "size" of the set $\mathbf{C}(P_{\mathrm{RT}}, \varepsilon)$ with Pr.

Corollary 6.2. *For every $M \in \mathbb{Z}^+$*

$$\mathrm{Pr}\left(\mathbf{C}\left(P_{\mathrm{RT}}, \frac{1}{M}\right)\right) = P_{\mathrm{RT}}\left(\mathbf{B}\left(P_{\mathrm{RT}}, \frac{1}{M}\right)\right) \leqslant \frac{1}{M}.$$

[3] A probabilistic anytime algorithm is different from a Monte Carlo algorithm. For such an algorithm there is no need of amplification as the quality measure is fixed a priori by the bound on the probability of error.

[4] $B_{P_{\mathrm{RT}},\varepsilon}$ is independent of x; recall that $\varepsilon = \frac{1}{M}$, $M \in \mathbb{Z}^+$.

Proof. As $\mathbf{C}(P_{\mathrm{RT}}, \varepsilon) = RT^{-1}(\mathbf{B}(P_{\mathrm{RT}}, \varepsilon))$, we have

$$\Pr\big(\mathbf{C}(P_{\mathrm{RT}}, \varepsilon)\big) = P_{\mathrm{RT}}\big(\mathbf{B}(P_{\mathrm{RT}}, \varepsilon)\big)$$

so from Corrolary 6.1 we deduce that for every $M \in \mathbb{Z}^+$ we have:

$$\begin{aligned}\Pr\left(\mathbf{C}\left(P_{\mathrm{RT}}, \frac{1}{M}\right)\right) &= P_{\mathrm{RT}}\left(\mathbf{B}\left(P_{\mathrm{RT}}, \frac{1}{M}\right)\right)\\ &= 1 - \mathrm{CDF}_{\mathrm{RT}}\left(\mathbf{q}_{\mathrm{RT}}\left(1 - \frac{1}{M}\right)\right).\end{aligned}$$

□

Comment. Corollary 6.2 is stronger than the ones obtained in [6,7] where the probability and the stopping decision depend on the program x.

To implement the anytime algorithm above we need an algorithm to compute

$$\mathbf{q}_{\mathrm{RT}}\left(1 - \frac{1}{M}\right) = \min\left\{t \in \mathbf{T}_{\mathrm{U}} \;\middle|\; \mathrm{CDF}_{\mathrm{RT}}(t) \geqslant 1 - \frac{1}{M}\right\}.$$

As the set $\mathbf{T}_{\mathrm{U}}$ is only computably enumerable, we will not be able to compute exactly $\mathbf{q}_{\mathrm{RT}}(1 - \frac{1}{M})$, but an upper bound for it. To this aim we consider a computably enumeration of $\mathbf{T}_{\mathrm{U}} = \{t_1, t_2, \ldots, t_i, \ldots\}$ and compute the following new bound:

$$\widetilde{\mathbf{q}_{\mathrm{RT}}}\left(1 - \frac{1}{M}\right) = \sum_{i=1}^{k} \rho(t_i) \geqslant \mathbf{q}_{\mathrm{RT}}\left(1 - \frac{1}{M}\right),$$

where

$$k = \min\left\{s \in \mathbb{Z}^+ \;\middle|\; \sum_{i=1}^{s} \rho(t_i) \geqslant 1 - \frac{1}{M}\right\}.$$

Obviously, the anytime algorithm will work correctly with the larger bound, but this will increase its time complexity.

7. Testing the quality of the running time distribution

Inference-based-decisions are made using statistical procedures based on sets of observations. An inference-based-decision of a *hypothesis* results in one of two outcomes: the hypothesis is accepted or rejected. The outcome can be correct or erroneous. The set of observations leading to the decision "reject the hypothesis" is called the *critical region*.

Fix the probability space $(A, \mathcal{B}(A), P_X)$ induced by a random variable X. Consider a critical region $B \subset A$, $B \in \mathcal{B}(A)$ and an observed value $x \in A$. For every $x \in A$, a *hypothesis* H_x is a statement such that "H_x is true" and "H_x is false" are measurable sets from $\mathcal{B}(A)$.

An inference-based-decision has the following form:

If the observed value $x \in A$ belongs to B, then decide to reject the hypothesis H_x.

An error occurs if we reject H_x on the basis of B, when H_x is true. The *probability of error*, that is, the probability of an erroneous decision, is $P_X(\{x \in B \mid \text{"}H_x \text{ is true"}\})$. Of course, only decisions with (very) low probability of error are of genuine interest.

Making the "right" choices for the running time computable probability distributions is essential for successful applications. There are a few ways to guide and improve the quality of these choices. One possibility is to test how "natural" is a particular computable probability distribution for some universal **U** using the sampling algorithm. For example we can use the two-sample Kolmogorov–Smirnov goodness-and-fit test (see [8, pp. 309–314]) to test how "natural" is a particular computable probability distribution, say P_ρ, for a given universal **U**.

We randomly sample $\mathbf{T}_\mathbf{U}$ to obtain a long sequence of independent, identically distributed running times $(t_1, \ldots, t_N)$ according to the discrete random variable RT. This can be achieved by a dovetailing method to generate sufficiently many L halting programs $\mathrm{POS}_L = (x_1, \ldots, x_L)$ and their running times $\mathbf{T}_{\mathrm{POS}_L} = (\tau_1, \ldots, \tau_L)$. Then we implement a random sampling (see, for example, [3,17]) to extract N identically distributed running times from $\mathbf{T}_{\mathrm{POS}_L}$, $(t_1, \ldots, t_N)$, which represent N independent, identically distributed replicates of the random variable RT.

We can now use the associated Empirical Cumulative Distribution Function defined by

$$\mathrm{ECDF}_{\mathrm{RT},N}(t) = \frac{\#\{1 \leqslant i \leqslant N \mid t_i \leqslant t\}}{N}, \quad t \in T_{\mathbb{U}}. \tag{13}$$

The two-sample Kolmogorov–Smirnov test compares these two empirical distribution functions in order to accept/reject *the null hypothesis* that the two datasets were drawn from "the same stochastic source". The null hypothesis, denoted by $H_0 : \{P_{\mathrm{RT}} = P_\rho\}$, states that the data produced by the sampling algorithm for the particular universal **U** fits the computable probability distribution P_ρ. The decision of accepting/rejecting H_0 is taken on the basis of numerical comparison of $\mathrm{ECDF}_{\mathrm{RT},N}$ and $\mathrm{ECDF}_{\rho,N}$.

8. Conclusions

In this paper we have proposed a probabilistic anytime algorithm for the Halting Problem. The anytime algorithm depends on the model of computation **U**; its quality depend on the computable probability distribution on the set stopping times.

The main motivation comes from experimental results reported in [27,30,31] and the theoretical results in [14, 15]: they all suggest that halting programs are not uniformly distributed and depend on the running times of the specific model of computation.

We have used the fact that programs which take a long time to halt stop at "algorithmically compressible times" [7] to construct a class of computable probability distributions on the set stopping times of halting programs. Each computable probability distribution induces a running time probability space on the set of halting programs, the probabilistic framework for our anytime algorithms.

Next we discuss some features of the proposed anytime algorithm. We start with positive features. (P1) The cut-off temporal bound does not depend on programs. (P2) The *a priori* class of computable probabilities distributions presented in Section 5.3 *are not arbitrarily pre-imposed*: they reflect the halting behaviour of the chosen universal machine through its running times. (P3) We can test empirically the choice of the computable probability distribution and sampling, hence adopt parameters suiting different universal machines.

However, the approach has limits. (L1) Because of (P1) and the use of a universal machine, the cut-off temporal bound could be large. This can be mitigated to some extent by (P3). (L2) Working with a fixed universal machine and programs x instead of pairs (program, y) increases the computational time as the simulation of the computation program(y) on **U** takes longer than running program(y).

It is a natural follow-up to study the computational complexity of the proposed anytime algorithms and, complimentary, to test experimentally their performance for classes of interesting programs.

Acknowledgements

We thank N. Allen, M. Holmes, Yu. Manin, L. Staiger and G. Tee for useful comments and suggestions. Special thanks to the anonymous referee for excellent critical remarks and suggestions that improved the paper. This work has been supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

References

[1] B.C. Arnold, N. Balakrishnan and N.N. Nagaraja, *A First Course in Order Statistics*, John Wiley, New York, 2008.

[2] L. Bienvenu, D. Desfontaines and A. Shen, What percentage of programs halt?, in: *Automata, Languages, and Programming I*, M.M. Halldórsson, K. Iwama, N. Kobayashi and B. Speckmann, eds, LNCS, Vol. 9134, Springer, 2015, pp. 219–230.

[3] K. Bringmann and K. Panagiotou, Efficient sampling methods for discrete distributions, *Algorithmica* (2016), 1–25. doi: 10.1007/s00453-016-0205-0.

[4] C.S. Calude, *Information and Randomness: An Algorithmic Perspective*, 2nd edn, Springer, Berlin, 2002.

[5] C.S. Calude and D. Desfontaines, Universality and almost decidability, *Fundamenta Informaticae* **138**(1–2) (2015), 77–84.

[6] C.S. Calude and D. Desfontaines, Anytime algorithms for non-ending computations, *International Journal of Foundations of Computer Science* **26**(4) (2015), 465–475. doi:10.1142/S0129054115500252.

[7] C.S. Calude and M.A. Stay, Most programs stop quickly or never halt, *Advances in Applied Mathematics* **40** (2008), 295–308. doi:10.1016/j.aam.2007.01.001.

[8] W.J. Conover, *Practical Nonparametric Statistics*, John Wiley, New York, 1971.

[9] B. Cook, A. Podelski and A. Rybalchenko, Proving program termination, *Communications ACM* **54**(5) (2011), 88–98. doi:10.1145/1941487.1941509.

[10] A. DasGupta, *Probability for Statistics and Machine Learning*, Springer, New York, 2011.

[11] R. Downey and D. Hirschfeldt, *Algorithmic Randomness and Complexity*, Springer, Heidelberg, 2010.

[12] C.A. Furia, D. Mandrioli, A. Morzenti and M. Rossi, *Modeling Time in Computing*, Springer, Berlin, 2012.

[13] J. Grass, Reasoning about computational resource allocation. An introduction to anytime algorithms, *Magazine Crossroads* **3**(1) (1996), 16–20. doi:10.1145/332148.332154.

[14] J.D. Hamkins and A. Miasnikov, The halting problem is decidable on a set of asymptotic probability one, *Notre Dame Journal of Formal Logic* **47**(4) (2006), 515–524. doi:10.1305/ndjfl/1168352664.

[15] S. Köhler, C. Schindelhauer and M. Ziegler, On approximating real-world halting problems, in: *Fundamentals of Computation Theory 2005*, M. Liskiewicz and R. Reischuk, eds, LNCS, Vol. 3623, Springer, 2005, pp. 454–466. doi:10.1007/11537311_40.

[16] R.H. Lathrop, On the learnability of the uncomputable, in: *Proceedings International Conference on Machine Learning*, L. Saitta, ed., Morgan Kaufmann, 1996, pp. 302–309.

[17] P.S. Levy and S. Lemeshow, *Sampling of Populations. Methods and Applications*, 3rd edn, John Wiley, NJ, 1999.

[18] M. Li and P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, 3rd edn, Springer Verlag, New York, 2008.

[19] N. Lynch, Approximations to the halting problem, *Journal of Computer and System Sciences* **9** (1974), 143–150. doi:10.1016/S0022-0000(74)80003-6.

[20] Y.I. Manin, *A Course in Mathematical Logic for Mathematicians*, 2nd edn, Springer, Berlin, 2010.

[21] Y.I. Manin, Renormalisation and computation II: Time cut-off and the halting problem, *Mathematical Structures in Computer Science* **22** (2012), 729–751. doi:10.1017/S0960129511000508.

[22] Y.I. Manin, Zipf's law and L. Levin probability distributions, *Functional Analysis and Its Applications* **48**(2) (2014), 116–127. doi:10.1007/s10688-014-0052-1.

[23] M. Minsky, *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1967.

[24] T. Mori, Y. Tsujii and M. Yasugi, Computability of probability distributions and distribution functions, in: *6th International Conference on Computability and Complexity in Analysis, Schloss Dagstuhl–Leibniz-Zentrum für Informatik*, A. Bauer, P. Hertling and K.-I. Ko, eds, Dagstuhl, 2009, pp. 185–196.

[25] P. Olofsson, *Probability, Statistics, and Stochastic Processes*, Wiley-Interscience, New York, 2005.

[26] A. Rybalov, On the generic undecidability of the halting problem for normalized Turing machines, *Theory of Computing Systems* **60** (2017), 671–676.

[27] F. Soler-Toscano, H. Zenil, J.-P. Delahaye and N. Gauvrit, Calculating Kolmogorov complexity from the output frequency distributions of small Turing machines, *PLoS ONE* **9**(5) (2014), e96223.

[28] E. Specker, Nicht konstruktiv beweisbare Sätze der Analysis, *The Journal of Symbolic Logic* **14** (1949), 145–158. doi:10.2307/2267043.
[29] K. Weihrauch, *Computable Analysis. An Introduction*, Springer, Berlin, 2000.
[30] H. Zenil, Computer runtimes and the length of proofs, in: *Computation, Physics and Beyond*, M.J. Dinneen, B. Khoussainov and A. Nies, eds, LNCS, Vol. 7160, Springer, 2012, pp. 224–240. doi:10.1007/978-3-642-27654-5_17.
[31] H. Zenil and J.-P. Delahaye, On the algorithmic nature of the world, in: *Information and Computation. Essays on Scientific and Philosophical Understanding of Foundations of Information and Computation*, G. Dodig-Crnkovic and M. Burgin, eds, World Scientific, Singapore, 2010, pp. 477–499.